实验1 类图绘制实验

**一、实验目的**

1. 学习类图的绘制；

2. 学习从系统需求中识别类，并构建相应的面向对象模型；

3. 学习使用顺序图描述对象之间的交互；

4. 学习使用活动图为业务流程建模；

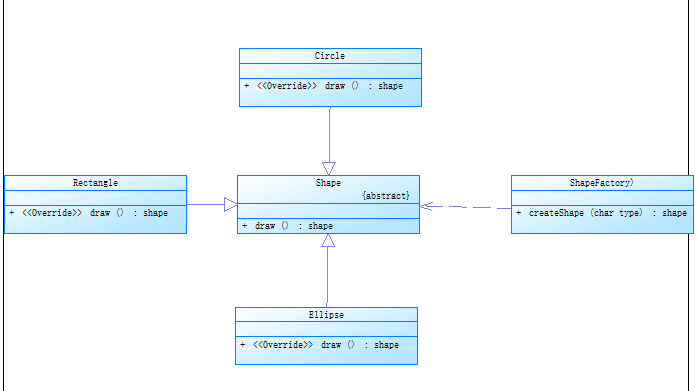
5. 学习使用PowerDesigner实现正向工程和逆向工程。

**二、实验内容**

1. 根据以下描述绘制类图，再正向工程生成Java源代码（也可生成其他面向对象语言的源代码，如C++或C#等）：

**图形(Shape)**可分为**圆形(Circle)、矩形(Rectangle)、椭圆形(Ellipse)**等具体图形，在Shape类中提供了一个**抽象的draw()方法**用于绘制图形，而在具体的图形类中实现该抽象draw()方法。

提供一个图形工厂类(ShapeFactory)，该类提供一个静态方法createShape(char type)，其返回类型为Shape，参数type为所需绘制图形对应的代码，例如“c”表示圆形，“r”表示矩形，“e”表示椭圆形等，在createShape()方法中，可以使用条件语句来判断所需绘制图形的类型，并根据参数的不同返回不同的具体形状对象。【注：“创建关系”是一种特殊的“依赖关系”】



2. 根据以下描述绘制类图：

某商场会员管理系统包含一个**会员类(Member)**，会员的**基本信息包括会员编号、会员姓名、联系电话、电子邮箱、地址**等，会员可分为金卡会员(GoldMember)和银卡会员(SilverMember)两种，不同类型的会员在购物时可以享受不同的折扣；每个会员可以拥有一个或多个订单**(Order)**，每一个订单又可以包含至少一条**商品销售信息(ProductItem)**，**商品销售信息**包括订单编号、商品编号、商品数量、商品单价和折扣等；每一条商品销售信息对应一类商品(Product)，**商品信息包括商品编号、商品名称、商品单价、商品库存量、商品产地**等。

3. 某基于C/S的即时聊天系统的注册和登录模块功能描述如下：

(1) 注册功能：用户通过注册界面(RegisterForm)输入新帐号，系统检测该帐号是否已存在，如果不存在则可注册成功，否则提示“帐号已存在”，用户再次输入帐号；用户输入其他个人信息；系统保存用户个人信息；用户个人信息（UserDTO）包括帐号、密码、姓名、性别、年龄、电话、电子邮箱等。

(2) 登录功能：用户通过登录界面(LoginForm)输入账号和密码，系统将输入的账号和密码与存储在数据库(User)表中的用户信息进行比较，验证用户输入是否正确，如果输入正确则进入主界面(MainForm)，否则提示“输入错误”。

现对这两个模块进行设计，要求如下：

(1) 根据以上描述**绘制类图**，要求分析出每个类中的主要方法；

(2) 需要提供独立的业务逻辑类和数据访问类，考虑到数据库的移植性，需提供抽象的数据访问接口；

(3) 尽量使用数据传输对象(DTO)来传递参数，减少方法中参数的个数。

**三、实验要求**

1. 熟练使用PowerDesigner绘制类图；

2. 熟练使用PowerDesigner实现正向工程和逆向工程；

3. 使用UML2.X中的标准图符绘制图形，对于一些较为复杂的图形，适当增加文字注释来进行说明；

4. 所绘制图形必须正确、完整，需具有较好的可读性，注意图形中图符和文字大小及布局。

**四、实验步骤**

1. 根据描述绘制类图并正向工程生成源代码；

2. 分析商场会员管理系统中实体类之间的关系，绘制相应的类图；

3. 分析某基于C/S的即时聊天系统的注册和登录模块的功能，绘制对应的类图；

**五、实验结果**

**(将绘制的类图截图 及 相关生成的代码 按题目序号粘贴在这里)**

**1.**

**Shape类：**

import java.util.\*;

/\*\* @pdOid 2b46445b-2c48-4c94-bdd6-5f49d2033fe3 \*/

public abstract class Shape {

/\*\* @pdOid 382f0742-7242-4c10-8a1b-162a850aa5b3 \*/

public void draw() {

// TODO: implement

}

}

**ShapeFactory类：**

import java.util.\*;

/\*\* @pdOid ae768e40-a310-4784-bc97-d7f51075b144 \*/

public class ShapeFactory {

/\*\* @pdOid 0f45a0c8-286e-47b0-9b58-3d3f7996be6a \*/

public Shape creatShape() {

// TODO: implement

return null;

}

}

**Retangle类：**

import java.util.\*;

/\*\* @pdOid 61680901-bb9b-4d69-907a-3fdbd3d3cfd6 \*/

public class Retangle extends Shape {

/\*\* @pdOid 48ab3358-3290-433e-8cd7-54ccaff55e6f \*/

public void draw() {

// TODO: implement

}

Ellipse类：

import java.util.\*;

/\*\* @pdOid d18dee9b-3ec3-48ea-96ca-04808e60fa9f \*/

public class Ellipse extends Shape {

/\*\* @pdOid 792b1d38-3291-439b-9d68-108327e26c8c \*/

public void draw() {

// TODO: implement

}

}

**Circle类：**

import java.util.\*;

/\*\* @pdOid ae57ad14-343b-4cee-a540-55c700c0b5e8 \*/

public class Circle extends Shape {

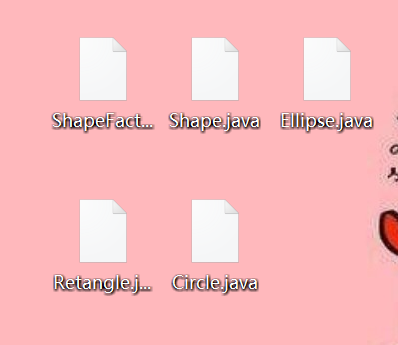
/\*\* @pdOid 73c1cb83-4c01-4db2-a264-29eff51c542e \*/

public void draw() {

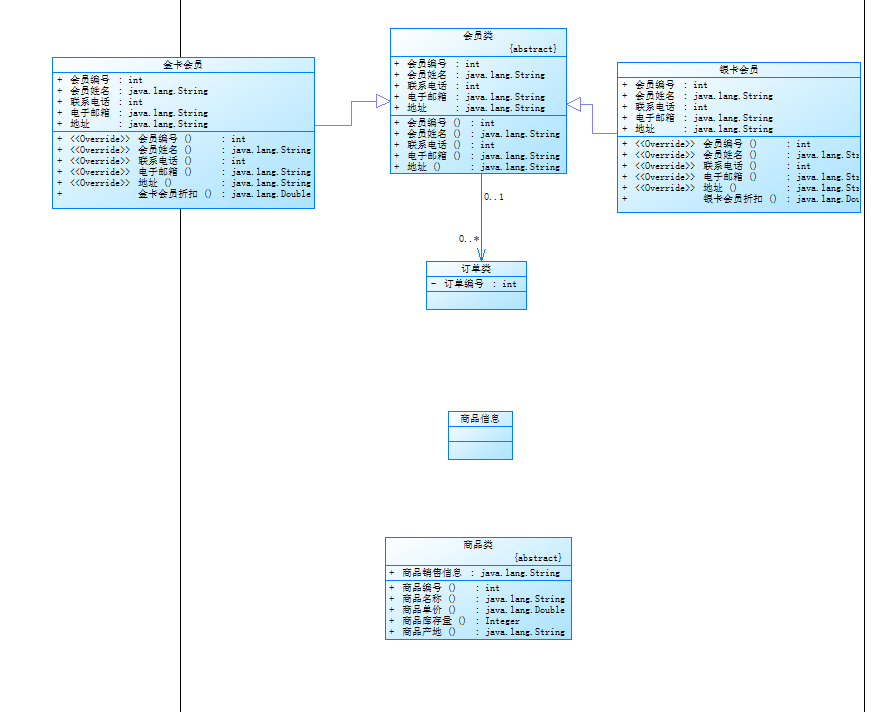
// TODO: implement

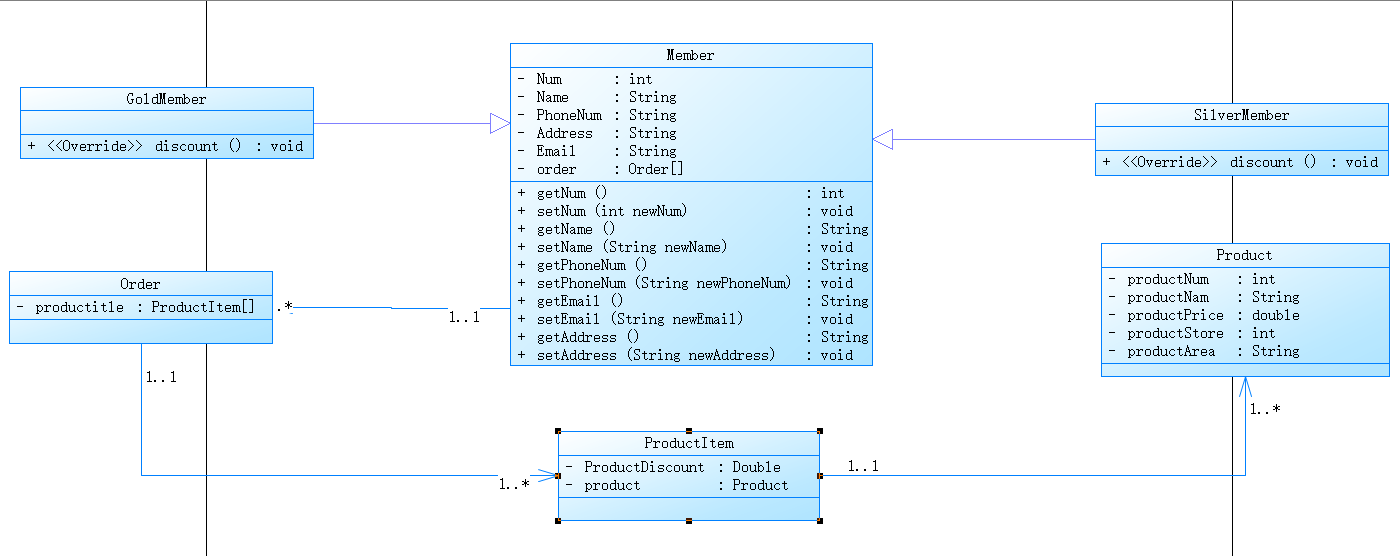
}

}

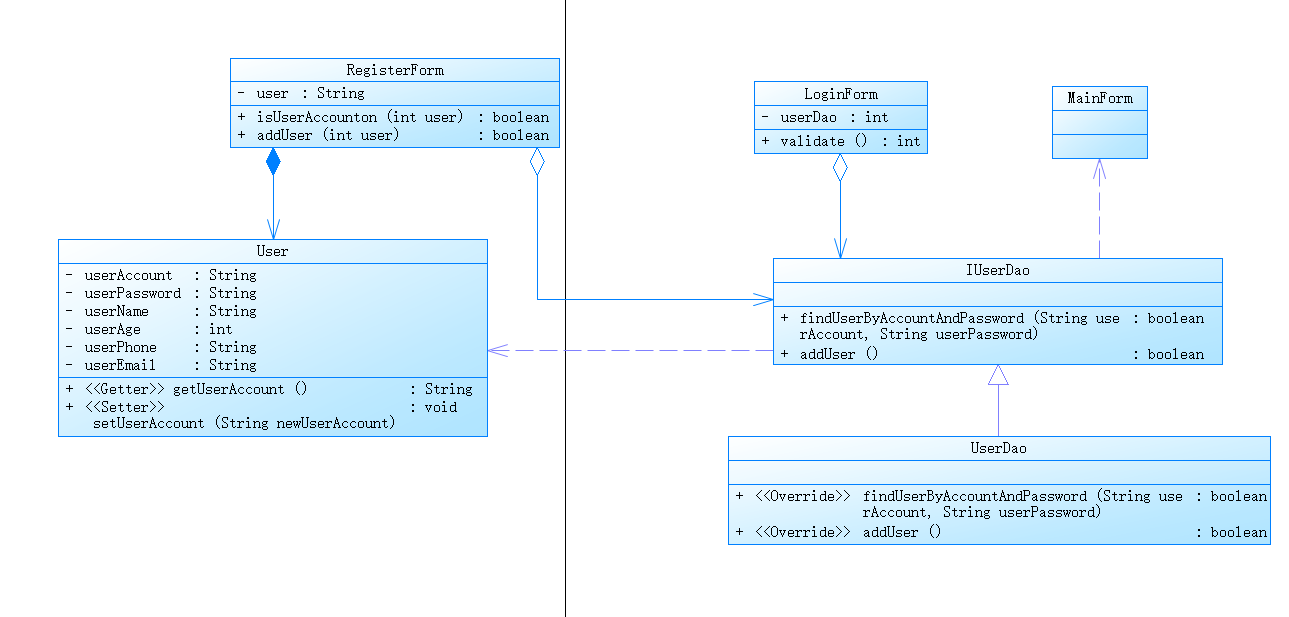


**2.**





**3.**



**六、实验小结**

这次实验，掌握了PowerDesigner的绘制类图操作，并且可以用类图通过正向工程生成类图的java代码。

再实验中，出现了不少的错误，总结如下几点：（1）.创建类之后，要建立静态方法，传入参数。（2）.构建简单工厂模式过程中，要注意抽象或者接口的建立，同时也不能忘记“工厂”的建立。（3）.建立类之间的关系要分清关系和箭头的方向。（4）.更要区分类名，方法名，参数名，避免语义上的混乱。